VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

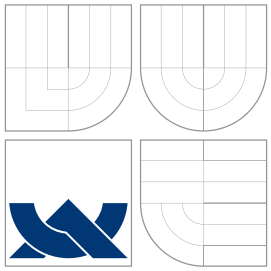
SHLUKOVÁNÍ SLOV PODLE VÝZNAMU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

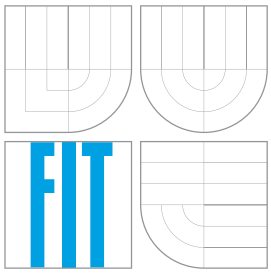
AUTOR PRÁCE
AUTHOR

JAKUB BÁRTA

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SHLUKOVÁNÍ SLOV PODLE VÝZNAMU

WORD SENSE CLUSTERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB BÁRTA

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2012

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací modulárního systému pro analýzu textového korpusu a následné vyhledávání sémanticky podobných slov. Systém umožňuje stemming korpusu, uživatel si může zvolit z různých způsobů analýzy korpus (matice spoluvýskytu, LSA).

Abstract

This bachelor's thesis deals with the design and implementation of a modular system focused on semantic similarity. System is able to stem the corpus and to analyze corpus in different ways - through coocurrence matrix or LSA.

Klíčová slova

přirozené zpracování jazyka, semantická podobnost, gensim, LSA, matice spoluvýskytu

Keywords

natural language processing, semantic similarity, gensim, LSA, coocurrence matrix

Citace

Jakub Bárta: Shlukování slov podle významu, bakalářská práce, Brno, FIT VUT v Brně, 2012

Shlukování slov podle významu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D.

.....
Jakub Bárta
14. května 2012

Poděkování

Na tomto místě chci poděkovat doc. RNDr. Pavlu Smržovi, Ph.D. za odborné vedení práce a konsultace. Dále patří velký dík mé rodině, která mě plně podporovala při studiu a bez které by to nešlo.

Obsah

Kapitola 1

Úvod

Shlukování slov podle významu je problematika, kterou se zabývá zpracování přirozeného jazyka (natural language processing – NLP). NLP se zabývá interakcemi mezi přirozenou lidskou řečí (psanou i mluvenou) a počítači. Cílem je, aby bylo možné komunikovat s počítačem pro člověka přirozeným jazykem, ne pouze skrz speciální jazyky a rozhraní. NLP se tedy zabývá jak pochopením a zpracováním jazyka, tak i jeho syntézou, tj. tvorbou pochopitelným a srozumitelných odpovědí.

V současné době je nutné zpracovávat čím dál větší objemy textů, ať už se jedná o vědecké články či lékařské zprávy, a orientace v nich se stává problematickou. Je proto žádoucí toto zpracovávání zautomatizovat pomocí počítačů.

Jedním ze způsobů takového automatického zpracování je analýza sémantické blízkosti slov – nakolik jsou si slova významově podobná. Sémantická blízkost se též nazývá vzdálenost nebo podobnost. Jedná se o metriku, která určuje podobnost jejich významů. Lze ji počítat jak pro slova, tak i věty a celé dokumenty. V případě vět a dokumentů pak určuje podobnost obsahu vět, resp. dokumentů. Blízká slova jsou například synonyma a antonyma.

Pro počítání sémantické vzdálenosti slov se využívá několik metod. Jedna z nich počítá podobnost jako nejmenší vzdálenost dvou uzlů v grafu v dané ontologii. Metody zjišťující podobnost slov na základě výskytu v textu jsou většinou založeny na statistické analýze rozsáhlých souborů textů, tzv. korpusů. Výpočet sémantické vzdálenosti nachází uplatnění např. v biomedicíně, v geografických expertních systémech, ve webových vyhledávačích a podobně. Pro vyhodnocování kvality výpočtu sémantické vzdálenosti se často využívají ručně vytvořené lexikální databáze, např. WordNet [4].

Mým úkolem v této práci bylo seznámit se se základy NLP, podrobněji prozkoumat způsoby analýzy korpusů a měření sémantické podobnosti slov a dokumentů. V další části jsem měl zpracovat a porovnat několik sad kontextů korpusu a navrhnout, implementovat a otestovat systém, který umožňuje vyhledávání sémanticky podobných slov.

Kapitola 2

Teorie

Kapitola obsahuje teoretický základ nutný pro pochopení této práce. Začíná objasněním pojmu korpus, pokračuje popisem kroků prováděných při automatické extrakci termínů (s detailním popisem metody tf-idf) a končí vysvětlením pojmů vektorový prostor a latentní sématická analýza a vztahu mezi nimi.

2.1 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka (NLP – natural language processing) je informatickým oborem využívajícím umělou inteligenci (obzvláště strojové učení [5]) za účelem zjednodušení komunikace mezi člověkem a strojem. Jedná se hlavně o proces získávání užitečných informací z přirozeného jazyka nebo jeho syntézu.

Jak již bylo řečeno, moderní algoritmy pro zpracování přirozeného jazyka jsou založené na strojovém učení, zejména statistickém strojovém učení. Orientace v této oblasti vyžaduje znalosti z lingvistiky, statistiky a dalších počítačových věd, jedná se tedy o značně mezioborovou problematiku.

Moderní metody zpracování přirozeného jazyka využívají strojového učení. V minulosti tomu však tak nebylo, NLP bylo řešeno kódováním velkého množství pravidel „natvrdo“ do zdrojového kódu. Paradigma strojového účení však preferuje jiný přístup – dává přednost obecným učícím se algoritmům. Tyto se používají k automatickému učení se pravidel z rozsáhlého korpusu dokumentů (viz 2.3).

Vysvětlíme to na příkladu určování slovních druhů. Úkolem je určit slovní druh všech slov ve větě (i neznámých slov). Typická implementace tohoto algoritmu založená na strojovém učení bude pracovat ve dvou krocích, učícím a odhadovacím. Učící krok využívá tréninkový korpus, který sestává z velkého množství vět, z nichž každá má ke každému slovu přiřazen správný slovní druh. Tento korpus je analyzován a je z něj vytvořen model složený z pravidel pro určování slovního druhu slova ve větě. V odhadovacím kroku je model použit ke zpracování nových (tj. nevyskytujících se v tréninkovém korpusu) vět. Důležitá část vývoje každého učícího se algoritmu je testování vytvořeného modelu na nových datech. Je velice důležité, aby se úspěšnost algoritmu netestovala na stejných datech, na jakých probíhalo učení. Jinak bude vycházet velice vysoká přesnost, která však nebude reálná.

V rámci NLP bylo uplatněno mnoho druhů učících se algoritmů. Všechny mají společné to, že na vstupu očekávají velké množství entit, které jsou získány ze vstupních dat. Jako příklad opět vezměme algoritmus pro určování slovních druhů – tyto entity pro něj mohou

být samotné určované slovo, slovo vlevo od něj, slovo vpravo od něj, slovní druh slova vlevo od něj atp.

Používané algoritmy se liší v druzích pravidel, které vytváří. Některé nejstarší algoritmy (např. rozhodovací stromy) vytváří soubor pravidel ve tvaru if-then, která jsou podobná těm „natvrdo“ zakódovaným do zdrojového kódu. Postupem času se algoritmy zakládají na statistickém modelu. Mají tu výhodu, že jsou schopné vyjádřit pravděpodobnost několika různých odpovědí, ne jen jedné, jak tomu bylo dosud. Takový výstup zajišťuje lepší výsledky v případě, kdy je algoritmus součástí většího systému. Dokonce se modely, které umožnují tento „měkký“ výstup, jeví jako robustnější a lépe se chovají v případě vstupu, který obsahuje chyby (což je u reálných dat běžné).

2.2 Sémantická podobnost

Jak již bylo řečeno, sémantická podobnost je v podstatě funkce, která každé dvojici dokumentů, resp. slov, přiřadí číselnou hodnotu v závislosti na jejich významové podobnosti [13].

2.2.1 Způsoby výpočtu sémantické podobnosti

Jedním ze způsobů je využití ontologií, na nichž se definuje topologická metrika (např. nejkratší vzdálenost mezi dvěma uzly v acyklickém grafu). Dalším je využití statistické analýzy a vektorového prostorového modelu, který je aplikován na vhodné textové korpusy a počítá podobnosti pomocí spoluvýskytu slov v textu.

Topologické metody

- hranové – jako zdroj dat jsou využity hrany a jejich typy, např. IntelliGO
- uzlové – zdroj dat jsou uzly a jejich vlastnosti, např. DiShIn

Statistické metody

Například:

- LSA – latent semantic analysis
- PMI – pointwise mutual information

2.3 Korpus

V lingvistice je pod pojmem korpus chápáno velké množství strukturovaného textu. Korpusy se používají pro statistickou analýzu, ověřování pravidel a další úlohy z oblasti NLP.

Korpus může obsahovat texty v jednom jazyce (monolingvní korpus) nebo vícejazyčné texty (multilingvní korpus).

Za účelem zlepšení použitelnosti korpusu pro NLP jsou korpusy anotovány. Anotace je proces, při kterém jsou do korpusu vloženy (většinou ručně) informace relevantní pro řešenou úlohu. Příklad anotace pro algoritmus určující slovní druhy je přiřazení správného slovního druhu každému slovu v korpusu ve formě značek (tagů).

Některé korpusy jsou strukturované ještě více do hloubky. Problémy spojené s tím, aby byl korpus plně a konzistentně anotován, omezují velikost korpusu na cca 1-3 miliony slov. Takové korpusy mohou být anotovány z hlediska morfologie, syntaxe i sémantiky.

Analýza a zpracování různých druhů korpusů je součástí výpočetní lingvistiky, rozpoznávání řečí a strojového překladu. Korpusy jsou často využívány ke konstrukci skrytých Markovových modelů pro určování slovních druhů a další účely. Korpusy lze považovat za pomůcku pro cizí (jazykově) pisatele. Znalosti gramatiky, které získá cizí pisatel prostudováním reálných textů v korpusu, mu umožní pochopit princip tvorby vět v cílovém jazyce a tak zlepšit jeho psaní.

2.4 Předzpracování textu

Předzpracování má za úkol upravit text tak, se dal snadněji zpracovat algoritmem pro výpočet sémantické vzdálenosti. Postup při předzpracování je popsán níže, přičemž v praxi užívané metody nemusí mít implementovány všechny části.

1. lexikální analýza
2. odstranění nevýznamových a nespecifických slov
3. lematizace a stemming
4. srovnání slov, resp. jejich kmenů nebo kořenů s termíny řízeného slovníku
5. vážení neboli stanovení vah termínů

2.4.1 Lexikální analýza

Problémy nastávají např. u zkratek, u nichž je třeba přesně identifikovat, zda je tečka součástí zkratky nebo plní pouze klasickou gramatickou funkci a ukončuje větu. Komplikaci představují rovněž slova se spojovníkem, u nichž je třeba stanovit, zda je daný výraz jednoslovný nebo zda bude chápán jako dvě samostatná slova. Další samostatná problematika se týká zpracování číslic. Zde je velmi důležité přesně určit, zda budou zpracovány jako závislé prvky, samostatná slova, či zda budou z dalšího zpracování úplně vypuštěny.

2.4.2 Odstranění nevýznamových a nespecifických slov

Nevýznamová jsou ta slova, která nenesou informaci, ale jsou součástí textu (např. spojky, částice apod.). Nespecifická slova se vyznačují vysokou mírou obecnosti, jež způsobuje, že je nelze chápat jako nositele pro daný dokument relevantní informace. Oba tyto druhy slov lze odstranit pomocí stoplistu[1].

2.4.3 Lematizace a stemming

Slova se v běžném textu vyskytují v různých tvarech. Pro zjednodušení zpracování je žádoucí tyto tvary redukovat na tvary základní, případně na kmeny nebo kořeny. Lematizace je proces vracející základní tvar slova, zatímco stemming vrací kmen daného slova [15]. Stemming se většinou provádí odsekáváním konců slov, takže tvary nepatřící k sobě mohou mít stejnou reprezentaci.

[1] Seznam slov, který se odstraní z dalšího zpracování. Může být zadán i regulárními výrazy.

Program, kterým se provádí lematizace, se nazývá lematizátor, programu či algoritmu provádějící stemming se říká stemmer.

2.4.4 Vážení

Slova v textu mají nestejnou důležitost pro reprezentaci obsahu dokumentu. Je tedy žádoucí stanovit relativní hodnotu zohledňující důležitost slova pro daný dokument. Obecně hovoříme o selektivní síle termínu, jež vyjadřuje schopnost termínu vyhledat z korpusu množinu dokumentů, která se bude lišit od množin vyhledaných pomocí jiných termínů [9]. Selektivní síla je nepřímo úměrná množství vyhledaných dokumentů.

Termíny lze vážit na základě několika základních parametrů, mezi které patří vlastní charakteristiky termínu (slovní druh, pád apod.), umístění termínu v textu (v případě vědeckého článku např. v abstraktu, závěru atd.) a frekvence termínu v textu (frekventovanější termíny v dokumentu jsou významnější než méně frekventované).

Tf-idf

Jedná se o způsob vážení termínů založený na frekvenci výskytu termínu v textu. Název vznikl z počátečních písmen anglických slov *term frequency – inverse document frequency*.

Je to číselný údaj vyjadřující důležitost slova v daném dokumentu v korpusu, který roste úměrně k množství výskytu slova v dokumentu a zároveň je kompenzován frekvencí výskytu slova v celém korpusu. Zohledňuje tedy skutečnost, že některá slova jsou běžnější než jiná [14].

Tf složka vyjadřuje, jak často se slovo nachází v dokumentu. Aby se zamezilo nadhodnocování dlouhých dokumentů, ve kterých se slovo vyskytuje častěji než v kratších, aniž by byl dokument relevantnější, probíhá většinou normalizace vydělením délkou dokumentu.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2.1}$$

kde $n_{i,j}$ je počet výskytů slova t_i v dokumentu d_j. Jmenovatel reprezentuje součet počtu výskytů všech slov v dokumentu d_j, tj. jeho délku.

Idf složka reprezentuje selektivní sílu slova. Čím častěji se slovo vyskytuje v dokumentech, tím menší selektivní sílu má (viz např. anglický neurčitý člen „a“ – vyskytuje se skoro ve všech anglicky psaných dokumentech, jeho význam pro reprezentaci obsahu dokumentu je však nulový).

$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|} \tag{2.2}$$

kde $|D|$ je velikost korpusu dokumentů, tedy počet dokumentů, ve kterých hledáme a $|\{j : t_i \in d_j\}|$ je počet dokumentů, které obsahují slovo i.

2.5 Reprezentace dokumentů ve vektorovém prostoru

Vektorový prostorový model je matematický model používaný k reprezentaci textových dokumentů jako vektorů.

2.5.1 Bag-of-Words

Bag-of-Words (dále BoW) je jeden ze zjednodušujících modelů používaných v NLP. V BoW je text reprezentován jako neuspořádaný soubor slov, tzn. nebere se ohled na gramatiku ani pořadí slov.

BoW je použit v mnoha metodách třídění dokumentů (např. je využit v bayesovských klasifikátorech). Využívá jej i LSA [12].

2.5.2 Princip

Vezměme v úvahu korpus obsahující dokumenty D_i, které jsou reprezentovány množinou termínů T_j (termíny mohou být váhované podle důležitosti, viz *tf-idf* 2.4.4) [8]. Každý dokument D_i je tedy definován n-rozměrným vektorem, kde n je počet termínů v množině T_j

$$D_i = (d_{i1}, d_{i2}, ..., d_{it}) \tag{2.3}$$

kde d_{ij} označuje váhu j-tého termínu.

Vektoru D_i se říká indexvektor daného dokumentu.

Ze dvou indexvektorů lze vypočítat podobnost příslušných dokumentů D_i a D_j, $s(D_i, D_j)$. Jako metrika se často používá *kosinová vzdálenost*, neboť je její výpočet méně náročný než výpočet úhlů svíraných ve vektorovém prostoru dokumenty D_i a D_j a osou a následný rozdíl těchto úhlů.

$$\cos\theta = \frac{D_i \cdot D_j}{\|D_i\| \, \|D_j\|} \tag{2.4}$$

kde D_i a D_j jsou indexvektory příslušných dokumentů a $\|v\|$ je velikost vektoru v

$$\|v\| = \sqrt{\sum_{i=1}^{n} v_i^2} \tag{2.5}$$

kde v_i je i-tý prvek vektoru v, který má n dimenzí.

Kosinová vzdálenost je pro identické dokumenty rovna jedné, pro naprosto rozdílné dokumenty je nulová.

2.5.3 Matice spoluvýskytu slov

Tato matice představuje jiný způsob reprezentace dokumentů ve vektorovém prostoru. Zahazuje informaci o tom, kde se termíny vyskytují, uchovává pouze informaci o tom, kolikrát se v daném kontextu dvojice termínů vyskytly.

Nechť je T množinou termínů, kde n je jejich počet. Matice spoluvýskytu má rozměry $n \times n$. Každý termín T_i je reprezentován n-rozměrným vektorem. Tyto vektory tvoří řádky matice spoluvýskytu. Na souřadnicích (i, j) leží hodnota udávající počet spoluvýskytů termínu T_i a T_j v daném kontextu.

Výhody vektorového prostorového modelu oproti standardnímu booleovskému modelu [7] jsou mnohé. Je to jednoduchý model založený na lineární algebře a je možné na něm provádět běžné vektorové operace. Umožňuje jemnější vážení než booleovský model a tedy i jemnější určování podobnosti.

Tento model má i svá omezení a nevýhody. Mezi nejvýznamnější patří podmínka, že slovo, pro které chceme určit slova podobná, musí přesně odpovídat termínu v dokumentu (hledání v podřetězcích často vyústí v nesprávnou shodu). Další nevýhodou je fakt, že dokumenty spojené se stejným tématem, avšak napsané jinou slovní zásobou, nebudou vyhodnoceny jako podobné (tzv. nesprávná neshoda). Dalším omezením vyplývajícím z definice vektorového prostorového modelu je ztráta informace o pořadí výskytu slov v dokumentu. Způsob vážení též představuje nepříjemnost – z intuitivního hlediska je v pořádku a dává smysl, není však formalizován. V současné době již jsou mnohá tato omezení překonána různými přístupy a metodami (např. SVD).

2.5.4 Zipfův zákon

Empirický zákon formulovaný G. Zipfem, který se týká statistické stránky lingvistiky [16]. Říká, že v rozsáhlých korpusech se nejčastěji se vyskytující slovo vyskytuje dvakrát tak často než druhé nejčastější slovo, třikrát tak často jako třetí nejčastější slovo atd. Jinými slovy, součin počtu výskytů (s) a pořadí slova (r) je konstantní.

$$r \cdot s = konst. \tag{2.6}$$

Zipfův zákon platí pro všechny jazyky a pro všechna slova, kromě těch nejméně obvyklých.

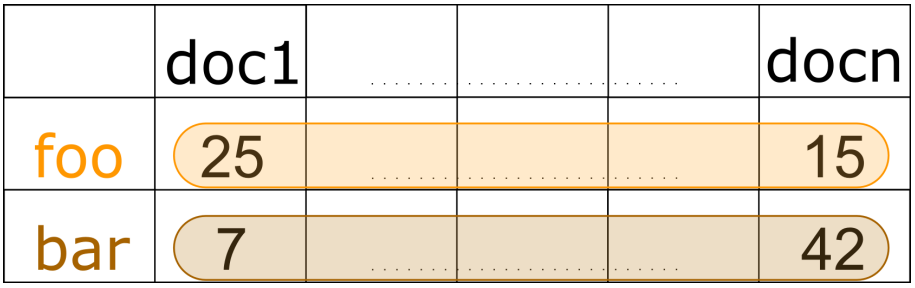
2.6 Latentní sémantická analýza

Latentní sémantická analýza (LSA) je metoda pro analýzu vztahů mezi dokumenty a slovy. Někdy se též označuje jako LSI (latent semantic indexing). Používá vektorovu reprezentaci textů (vychází tedy z vektorového prostorového modelu). Jejím hlavním účelem je výpočet sémantické blízkosti slov porovnáním příslušných vektorů [11].

2.6.1 Princip

LSA vytváří mnohodimenzionální vektorový prostor, ve kterém jsou slova reprezentována kontextovými vektory. Vektorový prostor je tedy tvořen maticí, kde řádky jsou kontextové vektory jednotlivých slov a sloupci jsou kontexty. Jako kontexty lze použít slova, kontextové okno různé velikosti[2] i celý dokument. Buňky matice vyjadřují, kolikrát se slovo vyskytlo v daném kontextu. V případě dokumentů je tato matice většinou normalizovaná, aby se předešlo nadhodnocování dlouhých dokumentů. Normalizace probíhá tak, že se každá buňka vydělí počtem slov v daném dokumentu. Vzhledem k faktu, že kontextové vektory jsou plnohodnotné vektory, je možné nad nimi provádět vektorové operace, mj. i porovnávání pomocí kosinové vzdálenosti, což bude klíčové pro výpočet podobnosti.

[2] Jako kontexty jsou brány úseky po x slovech, kde x je velikost okna



Obrázek 2.1: Ukázka matice výskytu s vyznačenými kontextovými vektory ke slovům *foo* a *bar*.

LSA rozšiřuje vektorový přístup použitím singulárního rozkladu (SVD – singular value decomposition – bude vysvětleno níže) a to hned ze dvou důvodů. Za prvé je pro rozsáhlé korpusy dimenze kontextových vektorů velká (korpusy mohou obsahovat až miliony dokumentů – dimenze vektorů by byly v řádech milionů), což je náročné na výpočetní výkon i paměť. SVD dovoluje tuto dimenzi velice výrazně snížit. Za druhé umožňuje z korpusu extrahovat latentní podobnosti a významy slov (tj. podobnost dvou slov založenou na výskytu ne spolu, ale v podobných kontextech, tzn. s podobnými slovy).

2.6.2 Singulární rozklad

Singulární rozklad je obecná maticová úloha definovaná takto:

„Nechť A je libovolná čtvercová matice. Pak existují ortogonální matice U a V a diagonální matice Σ, na jejíž diagonále jsou vlastní čísla matice $\sqrt{A^T A}$ tak, že

$$A = U\Sigma V^T \tag{2.7}$$

Výše uvedený rozklad se nazývá singulární rozklad vlastní čísla matice $\sqrt{A^T A}$ se nazývají singulární čísla matice A.“ [6]

Jak již bylo řečeno, SVD umožňuje redukovat dimenzionalitu kontextových vektorů a odhalovat latentní podobnost. Činí tak na základě reorganisace a přepočítávání kontextových vektorů. Jelikož dimenze vektorů při výpočtu SVD jsou řazeny od nejvýznamnějších po nejméně významné, dochází k zahození těch méně důležitých, což lze brát jako redukci šumu, chyb a nedůležitých souvislostí. Z praxe vyplývá, že pro zachycení významu textu postačuje 300 dimenzí (z původních desítek tisíc až milionů) [11]. Promítnutím kontextových vektorů do báze s nižší dimenzionalitou se docílí toho, že slova, která se vyskytují v podobných kontextech, budou mít podobné kontextové vektory a tudíž si budou blízká.

Hlavní nevýhodou SVD a tedy i LSA je výpočetní náročnost. SVD je velmi náročná operace. Singulární rozklad matice $m \times n$ má složitost $O(\min(mn^2, m^2n))$. Další nevýhodou je nutnost znovu počítat SVD při přidávání nových dokumentů do korpusu.

V současné době jsou již k dispozici metody, které obě nevýhody částečně eliminují, jejich popis je nad rámec této práce.

Kapitola 3

Návrh

Kapitola obsahuje popis průběhu návrhu - začíná definicí požadavků na systém, pokračuje architekturou a obsahuje též slepé vývojové uličky systému.

3.1 Požadavky na systém

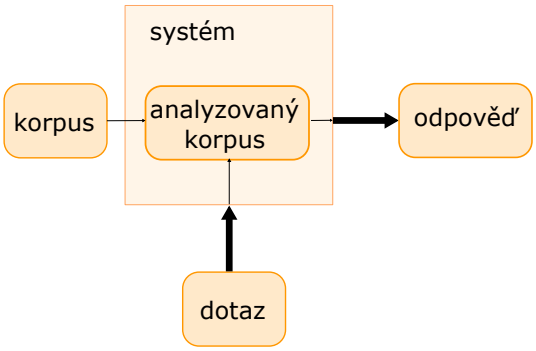
Systém musí být natolik modulární, aby umožňoval jednoduchou změnu formátu korpusu. Musí uživateli umožnit volit parametry analýzy korpusu – v kolika dokumentech se musí slovo vyskytovat, aby se bralo jako termín, v jakých kontextech chce uživatel vyhledávat podobnosti (celý dokument, kontextové okno).

Systém má být schopný zpracovávat korpusy různých velikostí (od jednotek MB do desítek GB).

Po zpracování korpusu je možné zadávat dotazy v podobě slov, systém na ně odpovídá výpisem nejpodobnějších slov.

3.2 Popis práce systému

Předtím než může systém začít odpovídat na dotazy ohledně podobnosti slov, je nutné analyzovat korpus, ze kterého bude informace o podobnosti čerpat. Poté, co je korpus analyzován, je možné systému pokládat dotazy, na které systém relevantně odpovídá.



Obrázek 3.1: Obecné schéma práce systému.

3.3 Automatická extrakce termínů

Pro automatickou extrakci termínů jsem původně chtěl využít práci Tomáše Strachoty [10], neboť zvládá i extrakci víceslovných termínů. Po jejím otestování na korpusu jsem však zjistil, že se pro můj účel nehodí – extrahované termíny byly příliš obecné a nepřesné pro daný dokument.

Rozhodl jsem se tedy využít knihovny *gensim*. Ta pro tento účel disponuje jednoduchým nástrojem umožňujícím jako termíny vyextrahovat slova, která se vyskytují ve více než nastaveném množství dokumentů korpusu. Tento způsob by však do termínů zařadil i členy, předložky apod. (a, the, of ...), což je nežádoucí. Z tohoto důvodu jsem použil stoplist a zavedl pravidlo, že termínem může být pouze slovo o minimální délce 2 znaky složené z písmen, přičemž může obsahovat pomlčku.

Stemming slov je možné zvolit, defaultní nastavení extrakce termínů je bez použití stemmingu.

Systém využívá dva druhy vážení – pro LSA se používá tf-idf, pro ostatní způsob, který je popsaný níže.

3.4 Analýza korpusu

Jak již bylo řečeno, systém musí umožnovat analyzovat korpus několika způsoby. Jako první způsob jsem zvolil tvorbu matice spoluvýskytu termínů pro kontextové okno o nastavitelné velikosti, druhý způsob je matice spoluvýskytu termínů pro celé dokumenty a poslední metodou je latentní sémantická analýza zprostředkovaná knihovnou *gensim*. Pro každý způsob je možné zvolit i variantu se stemmingem korpusu.

3.4.1 Matice spoluvýskytu termínů

Je to matice, kde řádky i sloupce jsou termíny a obsahem buněk je počet spoluvýskytů daných termínů v kontextu, pro který je matice vytvářena (kontext může být dokument, věta, odstavec, okno apod.). Prvky matice je vhodné nějakým způsobem vážit, protože např. hodnoty v matici tvořené pro okno o velikosti 3 slova a hodnoty v matici tvořené pro celý dokument se budou řádově lišit. Vážení provádím vydělením hodnoty na souřadnicích (x, y) počtem kontextů, ve kterých se slovo y vyskytlo.

3.5 Zjišťování sémantické blízkosti termínů

Pro zjištění sémanticky nejbližších termínů pro zadaný termín a kontext stačí projít příslušný řádek vážené matice spoluvýskytu a vybrat prvky s nejvyšší hodnotou. V případě použití LSA se nejbližší termíny zjistí výpočtem kosinové vzdálenosti.

3.6 Vyhodnocení výsledků

Původně jsem pro vyhodnocení výsledků systému chtěl použít FOLDOC [1], po shlédnutí výsledků jsem však seznal, že FOLDOC je pro tento účel nevhodný. Obsahuje hlavně definice pojmů, což je pro účel vyhodnocení nevhodné a termíny, které označuje jako blízké, jsou většinou natolik konkrétní, že se v systému vůbec nevyskytují (např. pro slovo „system“ FOLDOC vyhodnotí jako podobná tato slova: „sysop“, „Sysplex“, „sysprog“,

„System/360“, „System/370“, „System 5“, z nichž žádné systém neobsahuje (tj. žádné z nich nebylo vyhodnoceno jako termín)).

Rozhodl jsem se tedy výsledky vyhodnocovat následujícím způsobem – pro 25 náhodně vybraných vstupních termínů ručně zhodnotím, zda 10 nejpodobnějších výstupních termínů vypočítaných systémem je skutečně sémanticky spojených se vstupními termíny. Z takto vyhodnocených výstupů nakonec udělám shrnující statistiku.

Kapitola 4

Implementace

Tato část práce obsahuje popis implementačních detailů systému, které jsou nutné k porozumění jeho funkci.

Celý systém je implementovaný v jazyce *python*, který se díky své jednoduchosti a výborné podpoře práce s řetězci jeví jako nejvhodnější pro tento účel.

4.1 Požadavky na spuštění systému

Systém pro svůj běh potřebuje nainstalované tyto knihovny a programy:

- interpret jazyka *python v2.5* nebo vyšší
- knihovnu *gensim v0.8.4* pro python [2]
- knihovnu *nltk – natural language toolkit* pro python [3]

4.2 Korpus

V této práci vycházím z neanotovaného korpusu asi 124 000 vědeckých článků, který je dostupný na FIT VUT. Jedná se o články extrahované z PDF souborů dostupných na elektronických databázích odborných článků. Kvůli automatické OCR[1] extrakci textu se v článcích objevuje množství překlepů a chyb, ale vzhledem k velkému množství dokumentů v korpusu je možné tyto chyby zanedbat. Korpus je monolingvní, jeho jazykem je angličtina a je zaměřen hlavně na oblast počítačových věd.

4.3 Předzpracování korpusu

Pokud je při analýze korpusu vyžadován stemming, dochází nejprve k vytvoření stemmované verze korpusu a jejímu uložení na disk kvůli rychlejšímu zpracování.

Předzpracování tedy probíhá tak, že se načte obsah dokumentu (zajišťuje třída *Corpus*) a převede se na tokeny[2]. Poté jsou tokeny stemmovány a opět spojeny dohromady a uloženy na disk.

[1] OCR (Optical Character Recogniton) – automatický převod digitalizovaného textu na zpracovatelný počítačový text

[2] tokenizace – převedení na tokeny, tj. v tomto případě rozsekání v místech mezer a odstranění závorek, teček, čárek a uvozovek ze začátku a konce tokenu

```
# iterace nad celým korpusem
for file in corpus:
  tokens = tokenize(file)
  stemmed_tokens = []
  # postupné stemmování jednotlivých tokenů
  for token in tokens:
    stemmed_tokens.append(stemmer.stem(token))
  # a jejich spojení
  output = ' '.join(stemmed_tokens)
```

4.4 Automatická extrakce termínů

Automatická extrakce spočívá v postupné tokenizaci dokumentů, odfiltrování nevyhovujících tokenů (buď jsou příliš krátké nebo se vyskytují ve stoplistu – zajištěno třídou *CorpusFilter*, resp. *CorpusFilterNoStoplist*, která se používá pro stemmovaný korpus a která nepoužívá filtraci stoplistem) a předání vyhovujících tokenů instanci třídy *Dictionary*, která je součástí *gensimu* a vytváří slovník. *Dictionary* přiřadí každému termínu jednoznačný číselný identifikátor, který je pro pozdější práci vhodnější než textová reprezentace. *Dictionary* rovněž udržuje informaci o tom, v kolika dokumentech se dané slovo vyskytlo. Vytvořil jsem tedy slovníky od prahové hodnoty 100 až po 10 000 (skoky po 500) a ručně je analyzoval. Tímto postupem jsem zjistil, že vhodná prahová hodnota je 3100 dokumentů (slovník už neobsahuje málo významná a okrajová slova).

V případě stemmovaného korpusu je možné druhým průchodem vytvořit strukturu, která popisuje, které výrazy byly stemmovány na jaký kořen a kolikrát (např. na kořen „linguist“ bylo 25krát stemmováno slovo „linguistics“ a 12krát slovo „linguistic“).

```
# inicializace stromové struktury
tree = {}
for key in dictionary_stemmed.keys():
  tree[key] = {}

# iterace nad celým stemmovaným korpusem
for file in stemmed_corpus:
  tokens = tokenize(file)
  for token in tokens:
    stem = stemmer.stem(token)
    # pokud stemmovaná verze tokenu není ve slovníku ←
       termínů
    if not stem in dictionary_stemmed:
      # daný token nás nezajímá a pokračujeme dál
      continue
    # získání číselného identifikátoru stemmovaného tokenu←
       ze slovníku
    stem_id = dictionary_stemmed.token2id[stem]
    # pokud se token ještě ve stromu nevyskytuje
    if token not in tree[stem_id]:
```

```
        # přidáme ho
        tree[stem_id][token] = 1
      else:
        # pokud ano, zvýšíme počet stemmingů z daného tvaru ←
           o 1
        tree[stem_id][token] = tree[stem_id][token] + 1
```

4.5 Analýza korpusu

Jak již bylo řečeno, systém umožňuje 3 druhy analýzy korpusu. Prvním způsobem je tvorba matice spoluvýskytu termínů pro kontextové okno o nastavitelné velikosti, druhým je tvorba matice spoluvýskytu termínů pro celé dokumenty a třetím je latentní sémantická analýza. Následuje jejich podrobný popis.

4.5.1 Matice spoluvýskytu pro kontextové okno

Matice spoluvýskytu je popsána třídou *CoocurrenceMatrix*, což je v podstatě trojúhelníková matice (matice spoluvýskytu termínů je symetrická, proto ji lze redukovat na trojúhelníkovou). Tvorba této matice probíhá tak, že se postupně každý dokument tokenizuje a poté se zpracovává po kontextových oknech (např. pro kontextové okno +-2 se berou 2 tokeny před a po aktuálním tokenu, tj. celkem 5 tokenů).

Z kontextové okna jsou odfiltrována slova, která nejsou obsažena ve slovníku. Pro zbytek obsahu, tj. termíny, které jsou obsaženy ve slovníku, je veden počet kontextů, ve kterých se daný termín vyskytl (kvůli následnému vážení). Nakonec je pro každou dvojici termínů zvednuta hodnota spoluvýskytu v matici o 1 (reprezentující informaci, že se dvojice termínů opět vyskytla spolu).

```
matrix = CoocurrenceMatrix(len(dictionary))

occurence_cnt = {}
for key in dictionary.keys():
  occurence_cnt[key] = 0

for file in corpus:
  tokens = tokenize(file)
  position = window

  # každý průchod cyklu zpracuje jeden kontext
  while position < len(tokens):
    dictionary_words = []
    context = tokens[position-window:position+window]
    position = position + 2 * window

    # porovnání slov v okně se slovníkem
    for token in context:
      if token in dictionary.token2id:
```

```
        if dictionary.token2id[token] not in ↩
           dictionary_words:
          dictionary_words.append(dictionary.token2id[↩
             token])

    # aktualizace informace o počtu kontextů, ve kterých ↩
       se termín vyskytuje
    for word_id in dictionary_words:
      occurrence_cnt[word_id] = occurrence_cnt[word_id] + ↩
         1

    # zvýšení hodnoty spoluvýskytu v matici pro každou ↩
       dvojici termínů
    while len(dictionary_words) > 1:
      actual = dictionary_words.pop()
      for token in dictionary_words:
        if token == actual:
          continue
        else:
          matrix.increment(actual, token)
```

Po vytvoření matice spoluvýskytu je vhodné aplikovat vážení (viz 3.4.1). O to se stará třída *NormalizedCoocurrenceMatrix*, která vzniká spojením *CoocurrenceMatrix* a struktury zaznamenávající v kolika kontextech se jednotlivé termíny vyskytly (na tvorbu *NormalizedCoocurrenceMatrix* nelze použít pouze *CoocurrenceMatrix*, neboť ta neobsahuje informaci o tom, v kolika kontextech se termíny vyskytly). Z výše uvedeného vyplývá, že *NormalizedCoocurrenceMatrix* není realizována trojúhelníkovou maticí, nýbrž čtvercovou maticí.

Výsledkem vážení je tedy matice, jejíž prvky nabývají hodnot od 0 do 1 podle toho, jak podobné jsou si v rámci kontextového okna dané termíny.

4.5.2 Matice spoluvýskytu pro dokumenty

Principiálně velice podobné předchozímu případu, jen se jako kontext použije celý dokument.

4.5.3 Latentní sémantická analýza

Pro využití LSA k analýze korpusu je nutné nejprve zpracovat korpus *gensimem*. K tomu slouží třída *MmCorpus*.

```
  gensim_corpus = []

  for file in corpus:
    tokens = tokenize(file)
    # doc2bow je funkce, kterou se tokeny převádí do číselné↩
       podoby
    # a ta následně do Bag of Words representace
    gensim_corpus.append(dictionary.doc2bow(file_tokens))
```

```
# uložení gensim korpusu na disk
corpora.MmCorpus.serialize('...',gensim_corpus)
```

Po finálním zpracování (viz níže) metodou LSA však výsledky neodpovídaly očekávání. Pro kontrolu jsem vypočítal podobnost dvou totožných termínů, která měla vyjít jedna. Vyšla však jiná hodnota, usoudil jsem tedy, že tento způsob aplikace LSA bude chybný. Chybu se záhy podařilo objevit – *gensim* je vytvořen na počítání sémantické podobnosti dokumentů, nikoliv slov. Při dotazech na sémantickou podobnost slov tedy dává chybné výsledky. Řešením je následující transformace korpusu (v podstatě transpozice matice výskytu):

```
# načteme slovník a korpus
dictionary = corpora.Dictionary.load_from_text('...')
gensim_corpus = corpora.MmCorpus('...')

# inicializace transformační matice
matrix = {}
for key in dictionary.keys():
  matrix[key] = []


i = 0
# tvorba transformační matice
for file in gensim_corpus:
  for token in file:
    matrix[token[0]].append((i, token[1]))
  i = i + 1

texts = []

# transformace korpusu
for word_id in matrix.keys():
  tokens = []
  labels.append(dictionary[word_id])
  for tuple in matrix[word_id]:
    for count in range(tuple[1]):
      tokens.append(tuple[0])
  texts.append(tokens)

# tvorba nového slovníku a korpusu
transformed_dictionary = corpora.Dictionary(texts)
transformed_corpus = [dictionary.doc2bow(text) for text in
    texts]
```

Po této transformaci lze běžným způsobem korpus zpracovat dál:

```
termvectors = {}

# spočítáme tf-idf a vytvoříme wrapper okolo korpusu
tfidf = models.TfidfModel(transformed_corpus)
corpus_tfidf = tfidf[transformed_corpus]

# spočítáme LSA a vytvoříme wrapper
lsi = models.LsiModel(corpus_tfidf,id2word=dictionary, ↩
   num_topics=300)
corpus_final = lsi[corpus_tfidf]

# spočítáme vzájemné podobnosti termínů
index = similarities.MatrixSimilarity(corpus_final)
index.save('...')

# ještě je nutné uložit do struktury LSA reprezentace ↩
   jednotlivých
# dokumentů, neboť gensim neumožňuje přistupovat k ↩
   jednotlivým prvkům
# vypočítaného LSA, tzn. nelze snadno získat hodnotu
# corpus_final[word_id]
i = 0
for doc in corpus_final:
  termvectors[original_dictionary[i]] = doc
  i = i+1

save_termvectors()
```

Nyní stačí strukturu vytvořenou *gensimem* převést na třídu *NormalizedCoocurrenceMatrix*, aby se s ní dalo pracovat stejně jako s ostatními výslednými maticemi.

```
matrix = NormalizedCoocurenceMatrix(n = len(dictionary))

for word_id in dictionary:
  # funkce getLSAResults vrací výsledky LSA pro dané slovo
  results = getLSAResults(dictionary[word_id])

  for result in results:
    # nastavíme příslušnou buňku matice na hodnotu ↩
       vypočítanou LSA
    matrix.set(word_id, result[0], result[1])
```

4.6 Zjišťování sémantické blízkosti termínů

Protože jsou všechny výsledky všech druhů zpracování korpusu instancemi třídy *NormalizedCoocurrenceMatrix*, je možné k nim přistupovat stejně, a to funkcí *getWordSimilarities()*, viz níže.

```
matrix.getWordSimilarities(dictionary.token2id['...'], ←
    dictionary)
```

4.7 Přehled skriptů

Následuje přehled ovládacích skriptů se stručnými popisy:

create_dictionary.py z korpusu vytváří slovník termínů, umožňuje nastavit, v jakém minimálním počtu dokumentů se musí slovo vyskytovat, aby bylo považováno za termín a specifikovat stoplist

create_dictionary_no_stoplist.py identické s předchozím, jen bez stoplistu

create_corpus.py z korpusu a slovníku vytváří *MmCorpus*

stem.py pomocí *nltk* provádí stemming celého korpusu

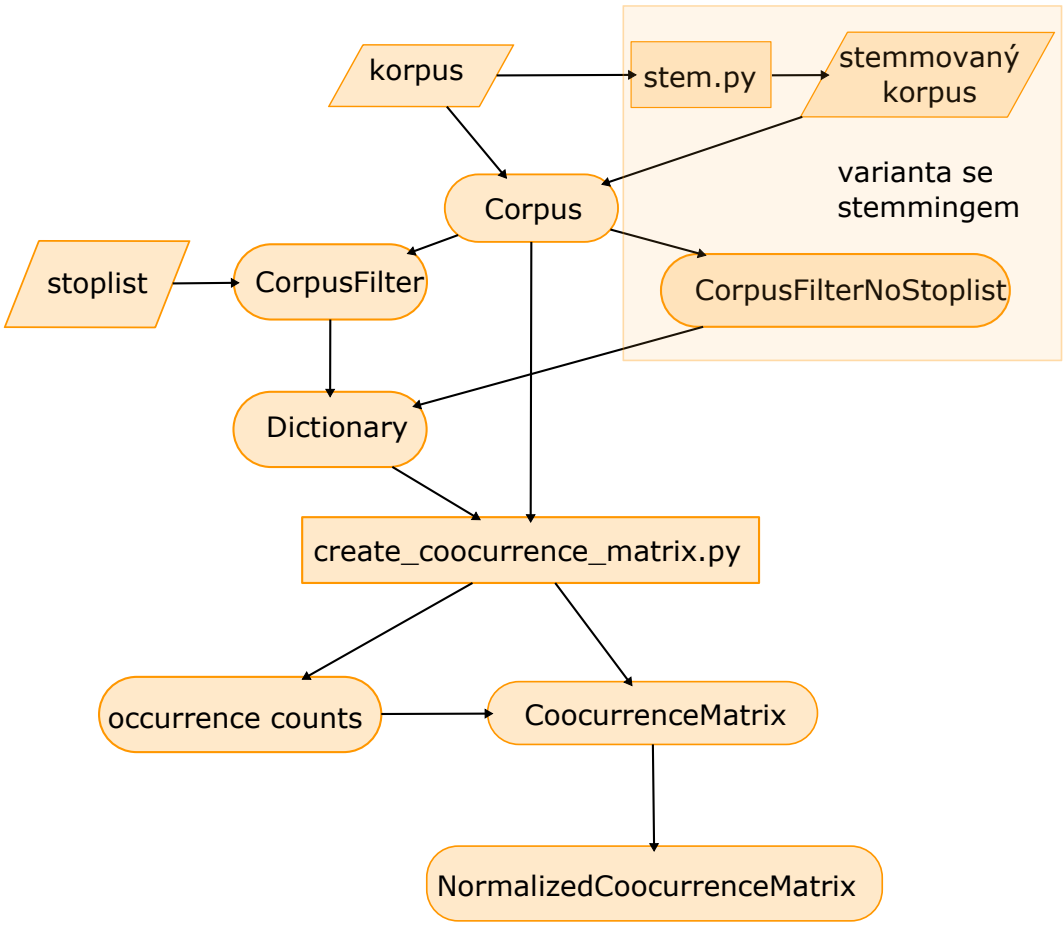
create_coocurrence_matrix.py v závislosti na parametrech vytváří příslušné matice spoluvýskytu

create_matrix.py transformuje *Dictionary* a *MmCorpus* tak, aby bylo možné počítat podobnosti dvojic slov, ne jen dokumentů

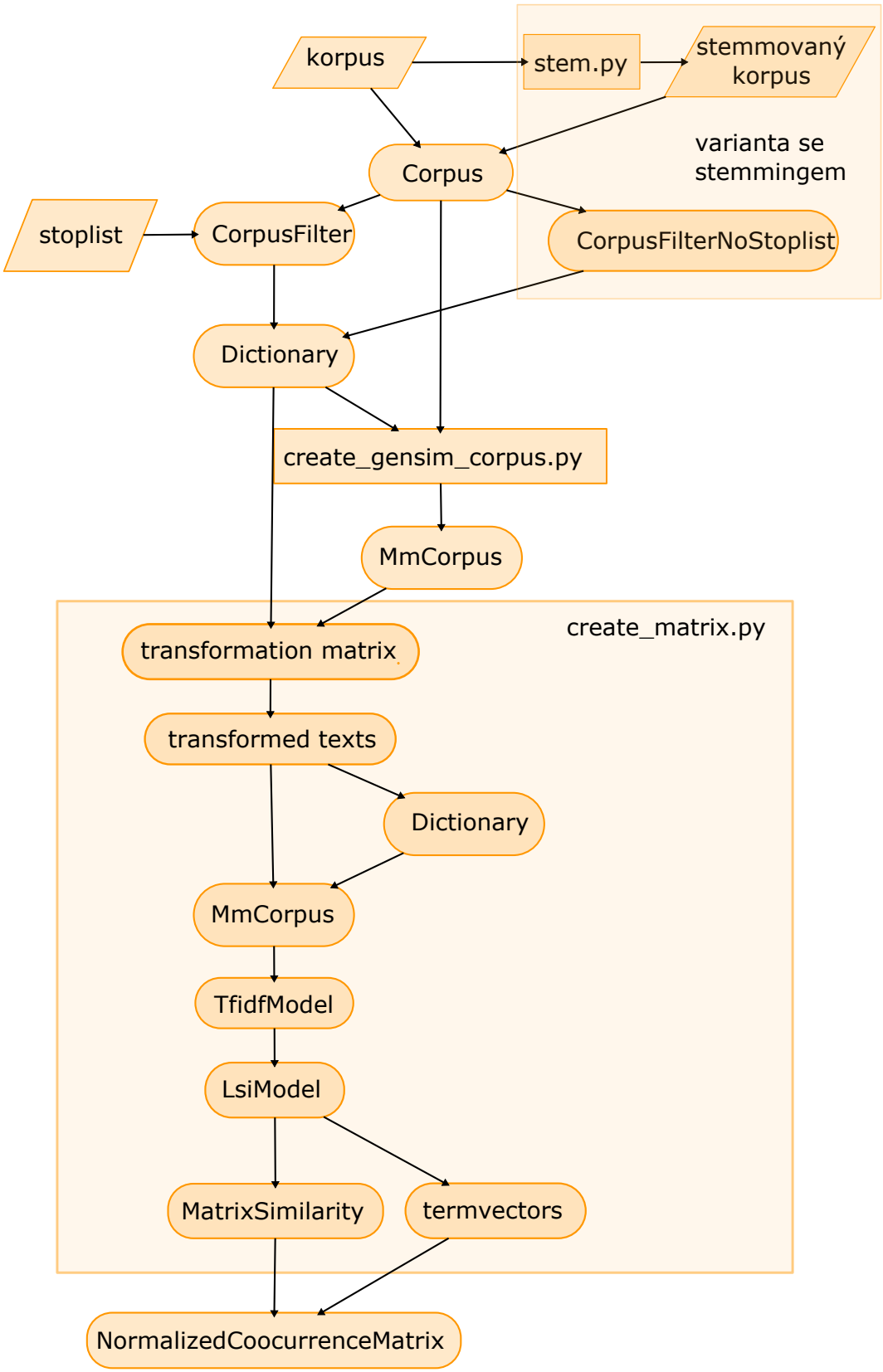
LSA_to_coocurrence_matrix.py převádí *MatrixSimilarity* na *NormalizedCoocurrenceMatrix*

4.8 Schémata práce systému

Následující schémata popisují práci systému pro výpočet LSA a spoluvýskytu slov. Kosodélníky představují entity vstupující do systému, obdélníky jsou skripty, ovály s počátečním velkým písmenem označují třídy. Ovály s malým počátečním písmenem jsou důležité proměnné.



Obrázek 4.1: Schéma práce systému pro výpočet spoluvýskytů.



Obrázek 4.2: Obecné schéma práce pro výpočet LSA.

Kapitola 5

Vyhodnocení výsledků

Tato kapitola obsahuje vyhodnocení výsledků systému, jejich shrnutí, zpracování a interpretaci.

V současné době není k dispozici thesaurus[1] ani anotovaný korpus zaměřený na počítačové vědy. Vyhodnocení výsledků je tedy nutné provést ručně, budou se vyhodnocovat výsledky všech způsobů zpracování korpusu – matice spoluvýskytu v kontextovém okně ±1 a matice spoluvýskytu v celém dokumentu. Pokus o výpočet LSA proběhl na 4 různých strojích. Výsledky LSA se nepodařilo získat, neboť nebyl k dispozici stroj, který by měl dostatečnou operační paměť pro výpočet transformační matice (viz 4.5.3).

5.1 Způsob vyhodnocení výsledků

Bylo náhodně vybráno 25 termínů z korpusu, pro každý termín bude vyhodnoceno 10 podle systému nejpodobnějších slov pro každý způsob zpracování (matice spoluvýskytu pro kontextové okno ±1 a matice spoluvýskytu v celém dokumentu). Vyhodnocením se rozumí rozhodnutí, zda je dané slovo skutečně podobné vstupnímu termínu na základě mých znalostí, případně výsledku vyhledávaní slova s termínem Googlem. Pokud není, je součástí vyhodnocení odůvodnění, proč se v 10 nejpodobnějších slovech vyskytlo.

fourier	coherent	expression	scalable	extracting
restriction	kernel	calibration	schema	laboratories
equipment	expense	electronics	male	diagnosis
indirect	calculus	method	elimination	maryland
tries	nominal	wavelet	powerful	model

Tabulka 5.1: Přehled náhodně vybraných termínů, které byly vyhodnocovány

První sloupec tabulky podobností obsahuje slovo, druhý vypočítanou podobnost se vstupním termínem a třetí poměr vypočítané podobnosti a průměrné podobnosti pro daný kontext (tento průměr je pro okno ±1 roven 0.00032310605604672171 a pro dokumenty 0.13273295866707982).

Pro větší přehlednost následuje podrobný popis vyhodnocení 10 termínů, pro zbylých 15 budou uvedeny pouze statistiky. Tabulky podobností pro zbylých 15 termínů jsou v příloze A.

[1] thesaurus – strukturovaný útvar obsahující slova seskupená podle významové podobnosti

short-time	0.1721088435	532.67
transform	0.1135661761	351.48
fractional	0.0748708599	231.72
transforms	0.0601857789	186.27
discrete	0.0509967758	157.83
fast	0.0410576816	127.07
series	0.0279696674	86.56
windowed	0.0275242241	85.19
transforming	0.0259163803	80.21
inverse	0.0201243146	62.28

Tabulka 5.2: *fourier*, okno ±1

dft	0.7538222337	5.68
fft	0.6768635044	5.1
short-time	0.6544300304	4.93
bandlimited	0.6264264264	4.72
time-frequency	0.6167554657	4.65
oppenheim	0.6111495047	4.6
nyquist	0.5730892183	4.32
convolution	0.560908637	4.23
aliasing	0.5541546093	4.17
windowed	0.5500646831	4.14

Tabulka 5.3: *fourier*, dokumenty

Všechna slova z levé tabulky (okno ±1) zcela jasně souvisejí s termínem *fourier*, jsou to vesměs přídavná jména označující varianty Fourierovy transformace (FT) (*short-time, fractional, discrete, fast, windowed, inverse*). Další slova jsou odvozena od tvaru *transform* (*transforms, transforming*). *Fourier series* je anglický ekvivalent Fourierovy řady, též tedy jasně souvisí s termínem *fourier*.

Pravá tabulka obsahuje zkratky spojené s FT – *dft, fft* (discrete Fourier transform, fast fourier transform), opět přídavná jména související s variantami FT (*short-time, bandlimited, time-frequency, windowed*). *aliasing, convolution, nyquist* jsou termíny, které souvisejí nebo využívají FT – podobnost prvních dvou je zřejmá, *nyquist* souvisí s Nyquist-Shannonovým vzorkovacím teorémem, který FT využívá. Jediné nesouvisející slovo je *oppenheim*, které je extrahováno z citací a vztahuje se na práce A. Oppenheima o FT.

---

automatically	0.0030376012	9.4
meaningful	0.001826484	5.65
fractional	0.0018231541	5.64
keywords	0.0014306152	4.43
reliably	0.0014035088	4.34
straight	0.0013808789	4.27
high-level	0.0013218099	4.09
repeatedly	0.0012507817	3.87
resonance	0.0012426648	3.85
visually	0.0011198208	3.47

Tabulka 5.4: *extracting*, okno ±1

extracts	0.3315412186	2.5
extract	0.2898427261	2.18
mining	0.2658086178	2.0
noun	0.2616734566	1.97
extraction	0.26028127	1.96
semantically	0.2567079463	1.93
extracted	0.2453608247	1.85
tags	0.2453416149	1.85
parsing	0.2420721375	1.82
textual	0.2410714286	1.82

Tabulka 5.5: *extracting*, dokumenty

Levá tabulka obsahuje slova, se kterými se termín často vyskytuje, sémanticky spojené jsou jen *automatically, keywords*.

Pravá tabulka obsahuje další tvary termínu (*extracts, extract, extraction, extracted*), které jsou významově spojeny s termínem. *Mining, parsing* má podobný význam jako termín, *noun, semantically, textual* však spojené s termínem není. *Tags* je též spojeno.

Důvodem horších výsledků (podle vypočítaných hodnot pravděpodobností i podle ručního vyhodnocení) u tohoto termínu je fakt, že se jedná o sloveso, navíc ještě v průběhovém tvaru.

imposed	0.005666181	17.54
imposes	0.004379186	13.55
severe	0.00403188	12.48
serious	0.0033910621	10.5
impose	0.0019807583	6.13
prevents	0.001862631	5.76
universal	0.0016903649	5.23
applies	0.0012697261	3.93
unnecessary	0.0010766194	3.33
guarantees	0.0009409993	2.91

Tabulka 5.6: *restriction*, okno ±1

restrictions	0.3127560776	2.36
predicate	0.2962616822	2.23
expressive	0.2884615385	2.17
restricting	0.279876161	2.11
notions	0.2676523069	2.02
atomic	0.2600855746	1.96
proving	0.2574572835	1.94
denitions	0.2544290606	1.92
closure	0.2517142857	1.9
corollary	0.2480641593	1.87

Tabulka 5.7: *restriction*, dokumenty

Impose, imposed, imposes, severe jsou slova velice často používaná spolu, dala by se považovat za ustálená sousloví. *Universal restriction* je pojem z popisové logiky. Ostatní slova mají s termínem žádnou nebo minimální podobnost.

Restrictions, restricting jsou tvary odvozené od termínu. *Predicate, atomic, closure, corollary* jsou slova spojená s termínem díky výrokové logice a důkazům. Ostatní nemají s termínem sémantickou spojitost.

---

xml	0.0310457967	96.09
conceptual	0.0154094694	47.69
relational	0.0130017762	40.24
evolution	0.0080602362	24.95
global	0.0043856885	13.57
database	0.0036636471	11.34
mappings	0.0031728514	9.82
matching	0.0031548649	9.76
annotation	0.0029785772	9.22
integration	0.0018882926	5.84

Tabulka 5.8: *schema*, okno ±1

xml	0.385345282	2.9
ontologies	0.3559171598	2.68
relational	0.3543890866	2.67
ontology	0.3242857143	2.44
expressive	0.2610340479	1.97
predicate	0.245482866	1.85
queries	0.2404542462	1.81
semantically	0.2391640867	1.8
semantics	0.2265077139	1.71
query	0.2243654822	1.69

Tabulka 5.9: *schema*, dokumenty

Všechna slova z levé tabulky ve spojení s termínem tvoří pojmy používané v počítačových vědách, dají se tedy považovat za sémanticky blízké.

Xml, ontologies, ontology, relational, semantically, semantics, query, queries jsou významově spojeny s termínem v oblasti databází a webu, *predicate, expressive* však blízké termínu nejsou.

murray	0.1704796191	527.63
bell	0.1517100263	469.54
telephone	0.0195736146	60.58
lincoln	0.0144025605	44.58
sharp	0.0092575948	28.65
research	0.0088629205	27.43
japan	0.0074836296	23.16
princeton	0.0057084035	17.67
national	0.0044056688	13.64
room	0.004244513	13.14

Tabulka 5.10: *laboratories*, okno ±1

murray	0.6308103661	4.75
hill	0.4745277619	3.58
bell	0.4566680515	3.44
staff	0.3762140733	2.83
president	0.3660968661	2.76
director	0.3492296405	2.63
chairman	0.3476342241	2.62
editorial	0.3462352585	2.61
juang	0.3205047319	2.41
fellow	0.3133235725	2.36

Tabulka 5.11: *laboratories*, dokumenty

Všechna slova z levé tabulky kromě *room* se často vyskytovala s termínem, jedná se o často používaná sousloví – názvy známých laboratoří apod. *Room* mezi ně však nepatří.

Pravá tabulka obsahuje i obecnější termíny spojené s termínem – *staff, president, director, chairman*. Poslední 3 slova (*editorial, juang, fellow*) však s termínem významově spojena nejsou. Jsou tedy vyhodocena jako chybná.

---

female	0.0356993519	110.49
speakers	0.0273982261	84.8
speaker	0.0088484993	27.39
native	0.0056265497	17.41
voice	0.004953099	15.33
subjects	0.0038745387	11.99
five	0.0037364984	11.56
seven	0.0028446221	8.8
six	0.0026196057	8.11
professional	0.0022155944	6.86

Tabulka 5.12: *male*, okno ±1

female	0.8338436959	6.28
unvoiced	0.4907581034	3.7
formant	0.4801561483	3.62
vowels	0.4795156408	3.61
speaker-independent	0.4780564263	3.6
consonants	0.463490099	3.49
vowel	0.4589698046	3.46
voiced	0.4574432577	3.45
speakers	0.4464671176	3.36
listeners	0.4295110094	3.24

Tabulka 5.13: *male*, dokumenty

V levé tabulce jsou *female, speakers, speaker, native, voice, subjects* spojeny s termínem, ale *five, seven, six, professional* byly systémem chybně vybrány jako podobné (chyba je pravděpodobně v nízkém množství výskytu termínu a tudíž velké ovlivnitelnosti výsledných podobných slov šumem).

Pravá tabulka obsahuje slova povětšinou spojená s akustikou a fonetikou, s přihlédnutím k povaze korpusu je tedy můžeme považovat za sémanticky podobná termínu.

model-based	0.007512816	23.25
medical	0.0059280254	18.35
clinical	0.0059075168	18.28
differential	0.0046361337	14.35
early	0.0023198012	7.18
failure	0.002007122	6.21
engine	0.0015675274	4.85
automated	0.0015333965	4.75
remote	0.0012164041	3.76
treatment	0.0011543424	3.57

Tabulka 5.14: *diagnosis*, okno ±1

diagnostic	0.3149628784	2.37
clinical	0.2960731816	2.23
patient	0.2884046274	2.17
tissue	0.2270401293	1.71
medicine	0.190026445	1.43
med	0.1844899507	1.39
medical	0.1676515826	1.26
health	0.1656671664	1.25
heart	0.1542870791	1.16
tomography	0.1501210654	1.13

Tabulka 5.15: *diagnosis*, dokumenty

Levá tabulka obsahuje většinou přídavná jména, která ve spojení s termínem tvoří často používaná sousloví nebo pojmy v lékařské vědě. Můžeme je tedy všechny považovat za sémanticky spojená s termínem (i *treatment*, které má zřejmou souvislost).

Pravá tabulka obsahuje obecnější slova spojená s lékařstvím, která jsou též všechna sémanticky spojena s termínem.

---

addressing	0.0117046445	36.23
branches	0.0048548499	15.03
costs	0.0028302894	8.76
eects	0.0028103617	8.7
branch	0.0023466667	7.26
matches	0.0014697684	4.55
alignment	0.0012113381	3.75
evidence	0.0011313202	3.5
sensing	0.000972668	3.01
discover	0.0009157509	2.83

Tabulka 5.16: *indirect*, okno ±1

noun	0.1291131307	0.97
predicate	0.1090342679	0.82
entity	0.103784023	0.78
textual	0.1016705069	0.77
addressing	0.1016242318	0.77
instruction	0.0994295029	0.75
intuition	0.0984825945	0.74
semantically	0.0972652219	0.73
accessed	0.0970528455	0.73
parsing	0.0958605664	0.72

Tabulka 5.17: *indirect*, dokumenty

Napřed je nutné vysvětlit slovo *eects* z levé tabulky. Jedná se ve pravděpodobně o slovo *effects*, které však bylo zkomoleno při OCR. Budeme tedy nadále pokračovat s nezkomolenou verzí.

Addresing, branches, branch, matches ve spojení s termínem tvoří sousloví používaná v počítačových vědách, zatímco *costs, effects, evidence, sensing, discover* tvoří sousloví obecná. Jediné nepodobné slovo je tedy *alignment*.

V pravé tabulce je vidět, že slova mají s termínem dokonce podprůměrnou podobnost. Z toho lze vyvodit, že výsledky budou nepřesné a chybné. Skutečně tomu tak je, jedinými slovy sémanticky spojenými s termínem jsou *adressing, instruction*.

proposed	0.0553976714	171.45
descent	0.0519523222	160.79
autocorrelation	0.032245019	99.8
utilizes	0.0296071635	91.63
outperforms	0.0295059469	91.32
novel	0.0283833369	87.85
exploits	0.0238983446	73.96
employs	0.0237295254	73.44
relies	0.022364959	69.22
indirect	0.0223156064	69.07

Tabulka 5.18: *method*, okno ±1

svd	0.881531154	6.64
regularization	0.8680732079	6.54
smoothness	0.866416197	6.53
intell	0.8543830494	6.44
med	0.8536215396	6.43
centroid	0.8535591094	6.43
descent	0.849980024	6.4
eigenvector	0.8488057684	6.39
amer	0.8472950697	6.38
high-resolution	0.8472622478	6.38

Tabulka 5.19: *method*, dokumenty

V obou tabulkách se nacházejí velmi nadprůměrné hodnoty podobností, lze tedy usuzovat, že výsledky budou správné a přesné.

Všechna slova v levé tabulce ve spojení s termínem tvoří sousloví velice často používaná v odborných článcích, hlavně při prezentování a vyhodnocování nových metod a postupů. Všechna je tedy můžeme považovat za sémanticky blízká termínu.

V pravé tabulce s termínem zřejmě souvisí *svd, regularization, med, centroid, descent, eigenvector, high-resolution*. *Smoothness* podobné termínu není a *intell, amer* jsou zkomolená slova, nelze je tedy považovat za podobná.

---

packet	0.0747039248	231.21
transform	0.061835783	191.38
transforms	0.059489566	184.12
bases	0.0505097118	156.33
coefcients	0.047221193	146.15
daubechies	0.0402870452	124.69
orthonormal	0.0358604785	110.99
decompositions	0.0345457312	106.92
coecients	0.0311929561	96.54
thresholding	0.0294295749	91.08

Tabulka 5.20: *wavelet*, okno ±1

daubechies	0.962033068	7.25
wavelets	0.8242728795	6.21
multiresolution	0.6568848758	4.95
subbands	0.6098310292	4.59
vetterli	0.6035621948	4.55
subband	0.5508792159	4.15
decompositions	0.4718110236	3.55
jpeg	0.4347053598	3.28
psnr	0.4218174376	3.18
multirate	0.4161684429	3.14

Tabulka 5.21: *wavelet*, dokumenty

Z hodnot v tabulkách je opět vidět, že výsledky by měly být správné. Je potřeba uvést na pravou míru slova *coefcients, coecients*. Jedná se o zkomolené verze slova *coefficients*.

Všechna slova z levé tabulky jsou sémanticky spojená s termínem přes oblast vln a vlnek, považujeme je tedy za sémanticky podobná termínu.

V pravé tabulce jsou s termínem spojena všechna slova kromě *vetterli*. Jedná se o jméno autora, který se do tabulky dostal pravděpodobně kvůli nízké frekvenci výskytu termínu a vysoké citovanosti jeho prací.

5.2 Shrnutí výsledků

Tato část obsahuje shrnutí výsledků vyhodnocení systému a jejich rozbor. Úspěšnost systému je měřena jako počet slov, která jsou v daném kontextu opravdu sémanticky blízká danému termínu.

Slovo	**Okno** ± 1	**Dokumenty**
calculus	8	9
calibration	4	3
coherent	10	7
diagnosis	10	10
electronics	8	7
elimination	8	4
equipment	10	6
expense	4	4
expression	8	8
extracting	2	7
fourier	10	9
indirect	9	2
kernel	9	9
laboratories	9	7
male	6	10
maryland	6	3
method	10	7
model	10	10
nominal	8	9
powerful	9	5
restriction	5	6
scalable	10	10
schema	10	8
tries	3	1
wavelet	10	9

Tabulka 5.22: Počty slov sémanticky spojených s termíny pro různé kontexty

Při ručním vyhodnocení dosáhl systém celkové úspěšnosti 73,20 %. Celková úspěšnost je průměrem úspěšností pro okno a dokumenty.

Okno ± 1	78,40 %
Dokumenty	68,00 %
Celkově	73,20 %

Tabulka 5.23: Úspěšnost systému pro všechny vyhodnocované termíny

Chyby byly nejvíce přítomny u sloves (viz 5.25) z toho důvodu, že slovesa nejsou častými nositeli významu a navíc náhodně vybraná slovesa pro vyhodnocení nebyla v základním tvaru. Podstatná jména (viz 5.24) a přídavná jména (viz 5.26) jsou mnohem lepšími a častějšími nositeli významu, čemuž odpovídají i výsledky.

Okno ±1	80,56 %
Dokumenty	71,67 %
Celkově	76,11 %

Tabulka 5.24: Úspěšnost systému pro vyhodnocovaná podstatná jména

Okno ±1	25,00 %
Dokumenty	40,00 %
Celkově	32,50 %

Tabulka 5.25: Úspěšnost systému pro vyhodnocovaná slovesa

Okno ±1	88,00 %
Dokumenty	54,00 %
Celkově	71,00 %

Tabulka 5.26: Úspěšnost systému pro vyhodnocovaná přídavná jména

Další chyby byly způsobeny zaměřením korpusu – některé okrajové, ne přímo počítačové termíny měly relativně nízkou frekvenci výskytu, byly tedy významně ovlivněny šumem a chybami v korpusu.

Ke správnosti výsledku nepřispěl ani způsob zpracování – OCR způsobuje mnoho překlepů, chyb a zkomolenin, které snižují úspěšnost.

Z vyhodnocení vyplývá, že vyšší úspěšnosti dosahuje kontextové okno ±1. Je to způsobeno faktem, že toto okno vyhodnocuje jako blízká ta slova, která tvoří častá sousloví s termínem a jsou tedy sémanticky spojena. Naproti tomu dokumentový kontext vyhodnocuje jako blízká ta slova, která se často vyskytují v dokumentu spolu s termínem, jedná se tedy o obecnější pojmy (např. *diagnosis* a *patient*, *method* a *svd*, *wavelet* a *psnr – peak signal-to-noise ratio*).

Je zajímavé povšimnout si faktu, že kontextové okno má tendenci hodnotit jako sémanticky podobná slova, která s termínem tvoří sousloví, tj. jakási klíčová slova či složené termíny. Dokumentový kontext naproti tomu hodnotí jako podobná slova, která se často vyskytují společně v dokumentu – tedy obecnější pojmy.

Kapitola 6

Závěr

Tato závěrečná kapitola shrnuje dosažené výsledky, diskutuje přínos práce a naznačuje směry vývoje a vylepšení.

6.1 Dosažené výsledky

V jazyce *python* byl s pomocí knihovny *gensim* implementován systém, který uživateli pomocí jednoduchých skriptů umožňuje analyzovat libovolný korpus. Díky modularitě systému stačí v rámci adaptace na nový korpus implementovat rozhraní pro čtení korpusu. Systém podporuji několik způsobů analýzy korpusu. Je možné využít stemming. Uživatel si může vybrat, pro jaký kontext chce podobnosti počítat – lze zvolit mezi pohyblivým oknem o nastavitelné velikosti, celými dokumenty a LSA. Po analýze korpusu lze systému zadávat vstupní termíny, pro které vyhledá nejpodobnější slova.

Systém bylo nutné otestovat ručně, neboť v současné době není k dispozici žádný thesaurus ani databáze podobných slov zaměřená na počítačové vědy. Ruční vyhodnocení tedy znamenalo ohodnocení, zda systémem vyhledaná slova jsou skutečně sémanticky spojená se zadaným termínem. Testování probíhalo na 25 náhodně vybraných termínech.

6.2 Přínos práce

V současné době žádný podobný systém, umožňující různé způsoby analýzy libovolného korpusu za účelem výpočtu sémantické podobnosti termínů, neexistuje. Přínosem této práce je tedy samotný systém a mj. i způsob, jakým lze adaptovat knihovnu *gensim* a její mocné transformační třídy na výpočet blízkosti dvojic slov, ne jen blízkosti dvojic dokumentů.

Systém je nyní nasazen na Fakultě informačních technologií Vysokého učení technického v Brně, kde se podílí na projektu *Decipher focused crawler*.

6.3 Možnosti dalšího rozvoje

Systém je otevřen mnoha úpravám a vylepšením – nejmarkantnější je asi zavedení podpory víceslovných termínů, což umožní zvýšení významové hodnoty termínů. Je též možné zavést vybírání termínů na základě slovních druhů, ne pouze minimálního počtu dokumentů, ve kterých se termín musí vyskytovat. Jako další vylepšení se nabízí využití knihovny *numPy* pro reprezentaci řídkých matic spoluvýskytu. Lze přidat vlastní implementaci latentní sémantické analýzy a osvobodit se tak úplně od knihovny *gensim*, případně přidat i jiné

transformace a způsoby výpočtu sémantické blízkosti (např. random indexing, latent Dirichlet allocation apod.). Dále lze přidat možnost výpočtu matice druhého řádu.

Literatura

[1] *Free On-Line Dictionary of Computing*. Dostupné na: <http://foldoc.org/>.

[2] *Gensim*. Dostupné na: <http://radimrehurek.com/gensim/>.

[3] *Natural language toolkit*. Dostupné na: <http://www.nltk.org/>.

[4] *WordNet*. Dostupné na: <http://wordnet.princeton.edu/>.

[5] Charniak, E. *Introduction to artificial intelligence*. [b.m.]: Addison-Wesley, 1984.

[6] Krátký, M. *Využití SVD pro indexování latentní sémantiky*. [Online; navštíveno 24.4.2012]. Dostupné na: <http://www.cs.vsb.cz/arg/techreports/lsi-svd_ma.pdf>.

[7] Lancaster, F. a Fayen, E. *Information Retrieval On-Line*. [b.m.]: Melville Publishing Co., 1973.

[8] Salton, G., Wong, A. a Yang, C. A vector space model for automatic indexing. *Communications of the ACM*. 1975, roč. 18. S. 613–620. Dostupné na: <http://portal.acm.org/citation.cfm?doid=361219.361220>. ISSN 00010782.

[9] Schwarz, J. *Současný stav a trendy automatické indexace dokumentů*. 2003. [Online; navštíveno 22.4.2012]. Dostupné na: <http://www.ikaros.cz/node/1300>.

[10] Strachota, T. *Extrakce klíčových slov z vědeckých článků*. Dostupné na: <https://merlin.fit.vutbr.cz/nlp-wiki/index.php/Rrs_keywords>.

[11] Wiemer Hastings, P. *Latent Semantic Analysis*. 2004. [Online; navštíveno 24.4.2012]. Dostupné na: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.6171&rep=rep1&type=pdf>.

[12] Wikipedia. *Bag of words model — Wikipedia, The Free Encyclopedia*. 2011. [Online; navštíveno 29.4.2012]. Dostupné na: <http://en.wikipedia.org/w/index.php?title=Bag_of_words_model&oldid=453620144>.

[13] Wikipedia. *Semantic similarity — Wikipedia, The Free Encyclopedia*. 2012. [Online; navštíveno 13.5.2012]. Dostupné na: <http://en.wikipedia.org/w/index.php?title=Semantic_similarity&oldid=492210012>.

[14] Wikipedia. *Tf*idf — Wikipedia, The Free Encyclopedia*. 2012. [Online; navštíveno 24.4.2012]. Dostupné na: <http://en.wikipedia.org/w/index.php?title=Tf*idf&oldid=488301207>.

[15] Wikipedie. *Stemming — Wikipedie: Otevřená encyklopedie*. 2012. [Online; navštíveno 23. 04. 2012]. Dostupné na: <http://cs.wikipedia.org/w/index.php?title=Stemming&oldid=8340121>.

[16] Zipf, G. K. *The Psychobiology of Language*. [b.m.]: Houghton-Mifflin, 1935.

Dodatek A

Tabulky podobností

První sloupec tabulky podobností obsahuje slovo, druhý vypočítanou podobnost se vstupním termínem a třetí poměr vypočítané podobnosti a průměrné podobnosti pro daný kontext (tento průměr je pro okno ±1 roven 0.00032310605604672171 a pro dokumenty 0.13273295866707982).

perfectly	0.010608049	32.83
narrow-band	0.007544757	23.35
semantically	0.0048516304	15.02
partially	0.0047606569	14.73
sources	0.0043328585	13.41
interference	0.003934739	12.18
wideband	0.003528435	10.92
spatially	0.0034672881	10.73
multipath	0.0031948882	9.89
fully	0.0030118737	9.32

Tabulka A.1: *coherent*, okno ±1

coherence	0.3250747544	2.45
wavelength	0.2908693275	2.19
aperture	0.2789446721	2.1
multipath	0.2699300699	2.03
beamforming	0.2698181818	2.03
steering	0.2668050869	2.01
bearing	0.240974212	1.82
antenna	0.2389643463	1.8
arrival	0.2369235531	1.78
antennas	0.2360335196	1.78

Tabulka A.2: *coherent*, dokumenty

closed-form	0.0711129522	220.09
regular	0.0274242549	84.88
analytical	0.024063252	74.47
referring	0.0147039688	45.51
analytic	0.0144124169	44.61
boolean	0.0126857557	39.26
explicit	0.0110802479	34.29
relating	0.0058457711	18.09
evaluates	0.0053636559	16.6
tractable	0.0050732807	15.7

Tabulka A.3: *expression*, okno ±1

expressions	0.5916932907	4.46
closed-form	0.5625	4.24
appendix	0.4615463285	3.48
expressing	0.4585414585	3.45
kronecker	0.4541995903	3.42
substituting	0.4434058464	3.34
calculus	0.4429429429	3.34
equality	0.4355173436	3.28
asymptotic	0.4340557276	3.27
right-hand	0.433201581	3.26

Tabulka A.4: *expression*, dokumenty

temporally	0.0067272116	20.82
video	0.0056261664	17.41
spatially	0.0053430342	16.54
highly	0.0038906053	12.04
codec	0.00372109	11.52
enabling	0.0035056968	10.85
coders	0.0033910095	10.5
bit-rate	0.0030760071	9.52
coder	0.0026624439	8.24
fully	0.0026353895	8.16

Tabulka A.5: *scalable*, okno ±1

scalability	0.5439494281	4.1
infrastructure	0.2160591653	1.63
client	0.215892908	1.63
delivery	0.2122958694	1.6
server	0.2100096246	1.58
packet	0.2089519167	1.57
heterogeneous	0.2010061743	1.51
codec	0.1902239345	1.43
mpeg	0.1837779574	1.38
progressive	0.1789702233	1.35

Tabulka A.6: *scalable*, dokumenty

hilbert	0.0182551883	56.5
smoother	0.010141706	31.39
fisher	0.0098910869	30.61
convolution	0.0068600934	21.23
gaussian	0.005296197	16.39
williams	0.0051530294	15.95
smoothing	0.0044489778	13.77
cubic	0.004314773	13.35
interpolation	0.0035138602	10.88
centers	0.0034729531	10.75

Tabulka A.7: *kernel*, okno ±1

kernels	0.74056353	5.58
bilinear	0.2638600652	1.99
time-frequency	0.2583134301	1.95
radial	0.2356828194	1.78
regularization	0.2280122013	1.72
hilbert	0.2108968473	1.59
separable	0.205988024	1.55
cohen	0.1983471074	1.49
cubic	0.1952	1.47
convolution	0.1942918955	1.46

Tabulka A.8: *kernel*, dokumenty

camera	0.0163024006	50.46
array	0.0032775011	10.14
intrinsic	0.0017630854	5.46
shape	0.0016377046	5.07
errors	0.0016223004	5.02
careful	0.0014100395	4.36
procedures	0.0014032783	4.34
curve	0.0012780656	3.96
geometric	0.0012115872	3.75
procedure	0.0011479173	3.55

Tabulka A.9: *calibration*, okno ±1

stereo	0.1853083434	1.4
camera	0.1629036868	1.23
wavelength	0.1585565883	1.19
perpendicular	0.1399435206	1.05
diameter	0.1328996283	1.0
microphones	0.1231307668	0.93
aperture	0.121670082	0.92
ray	0.1170777396	0.88
planar	0.1147946561	0.86
planes	0.11459833	0.86

Tabulka A.10: *calibration*, dokumenty

corporation	0.0085429142	26.44
measuring	0.0063622047	19.69
terminal	0.0058028617	17.96
recording	0.0053654147	16.61
maintenance	0.0027895559	8.63
radio	0.0025667582	7.94
commercial	0.0019527933	6.04
specialized	0.0018057785	5.59
audio	0.0017285141	5.35
electronic	0.0016644742	5.15

Tabulka A.11: *equipment*, okno ±1

tape	0.3012158055	2.27
microphones	0.2306713331	1.74
television	0.1950473861	1.47
electric	0.1879866518	1.42
facilities	0.1783235451	1.34
magnetic	0.1769642857	1.33
home	0.1733681462	1.31
engineer	0.170155902	1.28
pressure	0.1689269569	1.27
market	0.1682618092	1.27

Tabulka A.12: *equipment*, dokumenty

computational	0.0019778451	6.12
authors	0.0014335684	4.44
accounts	0.0012159903	3.76
characterizing	0.0006366385	1.97
imposes	0.0005151984	1.59
great	0.0004990643	1.54
editorial	0.0003763643	1.16
overhead	0.0003395586	1.05
probable	0.0003348401	1.04
slight	0.0003052814	0.94

Tabulka A.13: *expense*, okno ±1

reproduced	0.198325821	1.49
pay	0.1687622241	1.27
excess	0.1576086957	1.19
exceed	0.1562984663	1.18
sought	0.1511761331	1.14
policy	0.1510806762	1.14
savings	0.1488478306	1.12
title	0.1478121665	1.11
tradeoff	0.1476981003	1.11
reuse	0.1472222222	1.11

Tabulka A.14: *expense*, dokumenty

engineers	0.1519547954	470.29
iee	0.0221161096	68.45
letters	0.0208555921	64.55
massachusetts	0.0152105025	47.08
industrial	0.0108790252	33.67
government	0.0067938021	21.03
engineer	0.0065359477	20.23
naval	0.0064535054	19.97
corporation	0.0063872255	19.77
services	0.0053662759	16.61

Tabulka A.15: *electronics*, okno ±1

president	0.5133903134	3.87
editorial	0.5007559722	3.77
chairman	0.4920323609	3.71
engineers	0.4832314801	3.64
director	0.478601125	3.61
east	0.4494158588	3.39
educational	0.4430906389	3.34
staff	0.4346878097	3.27
permission	0.4330732293	3.26
professional	0.4218586843	3.18

Tabulka A.16: *electronics*, dokumenty

situation	0.0076549664	23.69
predicate	0.0067709427	20.96
relational	0.0043339254	13.41
event	0.0027509768	8.51
differential	0.0022946521	7.1
fractional	0.0019446977	6.02
pi	0.001706762	5.28
connection	0.001601794	4.96
logical	0.0013704719	4.24
proofs	0.0013507125	4.18

Tabulka A.17: *calculus*, okno ±1

predicate	0.2280373832	1.72
proving	0.202143064	1.52
expressive	0.1844262295	1.39
rst-order	0.1838637094	1.39
atomic	0.1708435208	1.29
notions	0.1672189651	1.26
closure	0.1594285714	1.2
completeness	0.1562583758	1.18
proofs	0.1561929119	1.18
symbolic	0.1472491909	1.11

Tabulka A.18: *calculus*, dokumenty

cut	0.0103003144	31.88
successive	0.0043103448	13.34
backward	0.003114591	9.64
redundancy	0.0021208908	6.56
gaussian	0.0017369576	5.38
iterated	0.0015752166	4.88
orders	0.0011937059	3.69
variable	0.0011776744	3.64
tag	0.001012878	3.13
highlight	0.0008563477	2.65

Tabulka A.19: *elimination*, okno ±1

eliminating	0.1789316399	1.35
eliminated	0.1576900412	1.19
eliminates	0.1439273553	1.08
eliminate	0.1414257086	1.07
proving	0.1230813785	0.93
calculus	0.1153153153	0.87
symbolic	0.114655571	0.86
completeness	0.1088180113	0.82
boolean	0.1086538462	0.82
predicate	0.1080996885	0.81

Tabulka A.20: *elimination*, dokumenty

phrases	0.0032754342	10.14
phrase	0.0012799895	3.96
attributes	0.0010964011	3.39
scales	0.001057306	3.27
resolutions	0.0009260688	2.87
predicate	0.0008922948	2.76
res	0.000880088	2.72
groups	0.0008790269	2.72
modification	0.000863906	2.67
bp	0.0008638376	2.67

Tabulka A.21: *nominal*, okno ±1

noun	0.199310561	1.5
linguistics	0.1166275501	0.88
parsing	0.1089324619	0.82
corpora	0.1088686202	0.82
lexical	0.1080324245	0.81
syntactic	0.1076058773	0.81
lexicon	0.1074168798	0.81
calibration	0.1071729958	0.81
phrases	0.1067028607	0.8
ambiguous	0.1066810345	0.8

Tabulka A.22: *nominal*, dokumenty

baltimore	0.0743034056	229.97
college	0.0565194181	174.93
park	0.0059778548	18.5
spring	0.0055544127	17.19
usa	0.0027494108	8.51
united	0.0026918998	8.33
avenue	0.0019603999	6.07
univ	0.0017228584	5.33
graduate	0.0013328002	4.12
md	0.000698812	2.16

Tabulka A.23: *maryland*, okno ±1

park	0.2014172472	1.52
md	0.1906496661	1.44
baltimore	0.18858084	1.42
college	0.1852902663	1.4
automation	0.1106639839	0.83
hopkins	0.1049101154	0.79
johns	0.1043890866	0.79
pennsylvania	0.0936957779	0.71
george	0.0882751151	0.67
army	0.0877137429	0.66

Tabulka A.24: *maryland*, dokumenty

agent	0.0005149846	1.59
policy	0.0004644898	1.44
repeatedly	0.0004169272	1.29
client	0.0003880816	1.2
encoder	0.000372755	1.15
engine	0.0003712565	1.15
proposal	0.0003664682	1.13
heuristic	0.0003471533	1.07
module	0.0003460367	1.07
en	0.0002740852	0.85

Tabulka A.25: *tries*, okno ±1

wants	0.2028361611	1.53
knows	0.1847472657	1.39
trying	0.1721763085	1.3
heuristics	0.1687318745	1.27
checks	0.1685069611	1.27
builds	0.1647058824	1.24
agent	0.1636582431	1.23
predicate	0.162305296	1.22
discover	0.1621140143	1.22
agents	0.1611611612	1.21

Tabulka A.26: *tries*, dokumenty

tool	0.0134877817	41.74
tools	0.0037758013	11.69
extremely	0.0037654965	11.65
potentially	0.0030592734	9.47
increasingly	0.0030108653	9.32
sufciently	0.0021687697	6.71
surprisingly	0.0021505376	6.66
mechanism	0.0021095905	6.53
sufficiently	0.0021061806	6.52
computers	0.0018501826	5.73

Tabulka A.27: *powerful*, okno ±1

expressive	0.3168348045	2.39
abstraction	0.2988990396	2.25
java	0.2859342916	2.15
relational	0.2835112693	2.14
predicate	0.2834890966	2.14
facilities	0.2768286172	2.09
functionality	0.2696800406	2.03
formalism	0.2674305217	2.01
infrastructure	0.2667723191	2.01
accessible	0.2663421419	2.01

Tabulka A.28: *powerful*, dokumenty

checking	0.1463172805	452.85
markov	0.0946974734	293.08
arma	0.0785482412	243.1
ar	0.0706945392	218.8
autoregressive	0.0511508951	158.31
mixture	0.0499090891	154.47
language	0.0427855813	132.42
conceptual	0.0410090717	126.92
parametric	0.0393188738	121.69
sinusoidal	0.0364702517	112.87

Tabulka A.29: *model*, okno ±1

hmms	0.9615307953	7.24
hmm	0.9450211457	7.12
markov	0.9111795019	6.86
model-based	0.9084636615	6.84
posterior	0.9079908199	6.84
mixtures	0.9067778936	6.83
modeled	0.8902904645	6.71
modelled	0.8893023256	6.7
mixture	0.8860378048	6.68
viterbi	0.8851612903	6.67

Tabulka A.30: *model*, dokumenty